C O M M O D O R E

PET USERS' CLUB NEWSLETTER

ISSUES 1 and 2

ooOoo

INDEX

COMMODORE PET USERS CLUB

NEWSLETTER

Commodore would like to apologise for the delay in producing the No. 1 Newsletter. We decided to combine numbers 1 and 2 in one bumper issue!

If you would like a copy of issue number Ø (APRIL) you may elect to back date your membership to a particular issue such as this. If so please write and advise us. Your membership will be adjusted accordingly and any relevant issues will be forwarded to you.

Included at the back of this issue are details of products marketed by non-Commodore companies which may be of interest to you. The inclusion in our newsletter is neither a recommendation or otherwise but are included as items of interest to PET users.

WOULD YOU LIKE TO BE A PET VET?

Commodore Systems are looking for a bright young engineer to assist with servicing PETs and also after some instruction field some technical enquiries. The ideal person would be aged between 18 and 21 and have some practical experience with and good theoretical knowledge of 68ØØ type microprocessors.

If you are interested please get in touch with:

- Commodore Systems,
360 Euston Road,
London. NW1.
Tel. 388 5702.

As Commodore Systems is growing rapidly vacancies may occur for other technical, software and sales personnel. If you would be interested in working with PET at a future date please send us a curriculum vitae and we will hold it on our files - confidentiality respected.

THE MASTER LIBRARY and

THE COMMON LIBRARY

There has been a little confusion regarding what you get when you submit a tape to Commodore. What happens is as follows: If your program is judged to be of high enough quality and general interest, you will receive the sum of £25.-, your program will be entered into the official Commodore MASTER LIBRARY and becomes the sole property of Commodore. If the program is accepted but not suitable for the MASTER LIBRARY, you will receive a form enabling you to request up to four programs from, and your program will be entered into the Users Club COMMON LIBRARY. If you wish to purchase programs from the COMMON LIBRARY directly a charge of £1.25 is made per program with a minimum order value of £5.- (postage and packing inc.) Commodore will not return any cassettes submitted so please make sure you have a copy. Programs submitted to Commodore must be fully documented on tape. This means that anybody wishing to use this program must be able to do so from loading and running the program only, not from any instruction sheet.

COMMODORE

Commodore Systems Division
360 Euston Road, London NW1 3BL
Telephone 01-388 5702

Official Price List (April 1978)

Hardware	Retail Exc. VAT	Retail Inc. VAT
PET 2001-8 Personal Computer	£643.52	£695.00
PET 2nd Cassette Deck (June release)	55.00	59.40
PET 2020 Printer (Aug. release)	425.00	459.00

Accessories (cash terms only)

PET Users Handbook	-	5.00
PET Introductory Booklet	-	1.00
6500/KIM Programming Manual	-	5.00
6500/KIM Hardware Manual	-	5.00
KIMI User Manual	-	5.00
PET Users Club Membership	-	10.00
Special C12 high quality blank cassettes (minimum quantity 10)	-	.50

MASTER LIBRARY (cash only)

Othello	Cunning game of skill. Two levels of play, you against the computer.	£ 8.00
Pontoon	Board game. True 52 card pack plus amazing graphics.	6.00
Wrap-Trap	Dynamic graphics game in which the player has to trap the computer. Good arcade quality graphics.	8.00
Noughts and Crosses	You against the computer.	3.00
Lunar Lander	First class game of skill - in real time and with the odd surprise !	8.00
Rotate	Difficult if you are not an expert ! Similar to little plastic trays with moveable letters and letter missing.	5.00
Biorhythms	Carefully written programme. Good graphics, with a real plot routine.	8.00
Disassembler	6500 series full disassembler asks for decimal starting location and lists from this point, gives full mnemonics and handles ASCII tables.	15.00
Machine Code Handler	This programme allows you to type in a list of HEX codes from a given location. These routines can then be called using the SYS verb.	3.00

PLEASE NOTE There is a minimum order value of £5.00

Planned Introductions during 1978

A floppy disc, memory expansion and modem are planned for introduction later in 1978. Additional programmes will be introduced at regular intervals including commercial, scientific and statistical programmes.

Directors: I. Gould (Chairman), J. Tramiel (Managing), C. T G Fish, C. Spencer, R. Gleadow
Reg. Office: Industrial Estate, Eaglescliffe, Stockton-on-Tees, Cleveland. TS16 0PN Reg in England Reg. No 956774

SECOND CASSETTE DECKS

.... are now available if you wish to order them

at a cost of £55.- plus VAT. (Cash with order please.)

COMMON LIBRARY

Here is the COMMON LIBRARY listing so far:-

ØØ1	ESP TEST - submitted by	Mr. Chambers Co. Mayo Ireland
ØØ2	SLOT MACHINE	" " "
ØØ3	MASTERMIND -	Mr. McDonald - Brighton
ØØ4	MOO -	Prof. A.Colin - Univ. of Strathclyde
ØØ5	LIFE -	Mr. Wheatcroft - Watford
ØØ6	STARWARS -	Dr. Lucas - Univ. of Manchester
ØØ7	1 ARM BANDIT -	A.M. Robertson - Holland
ØØ8	DEEPSPACER -	Mr. DA Allen, Colchester Essex.
ØØ9	SOLVING SIMULTANEOUS EQUATIONS -	Prof. Colin - Univ. of Strathclyde
Ø1Ø	LEAST SQUARES -	Mr. Clintworth - Imperial Coll. London.
Ø11	MEMORY DISPLAY IN HEX -	Mr. Tribe, Gwent

Minimum order four programs £5.ØØ

each additional program is £1.25 p+p inc.

PLEASE ORDER BY <u>PROGRAM NUMBER</u>

65Ø2 JOURNAL & COMPUTABITS

It seems that the 65ØØ series microprocessors are becoming so popular that they now warrant a journal of their own. 65ØØ series devices are used in the PET, the APPLE, OHIO SCIENTIFIC and numerous other top selling machines. This journal is called "MICRO" and has a single copy price of £1.7Ø and an annual subscription of £9.ØØ for six issues. I have had a good look at an issue myself and find it excellent. It is obtainable from The Computer Book Shop, Temple House, 43-48 New Street, Birmingham. B2.

Another excellent publication, this time home grown, is Computabits. This is a generally small systems orientated publication often carrying information about the PET and is very professionally produced. This publication is obtainable from Computabits Ltd., 41 Vincent Street, Yeovil, Somerset. Copies are 85p each. Annual subscription is £5.ØØ for approximately six copies.

LP ENTERPRISES are offering 5% discount to all PET USERS CLUB MEMBERS. (see list on following page).

STOP PRESS Computabits inform us that they are in a position to supply an interface board to allow the PET to drive ordinary video monitors etc. Price on application.

LP ENTERPRISES

313 Kingston Road, Ilford,
Essex, IG1 1PJ England.

From the representatives in Europe . . . for America's leading Micro-computer magazines and books, for the hobbyist, educationalist and professional alike, we bring you a little light reading!

From Adam Osborne Associates

INTRODUCTION TO MICROCOMPUTERS	
Volume 0: The Beginners Book	£5.95
Volume 1: Basic Concepts	£5.95
Volume 2: Some Real Products (Revised Late 1977)	£11.95
6800 Programming for Logic Design	£5.95
8080 Programming for Logic Design	£5.95
Z80 Programming for Logic Design (Available March 78 approx)	£5.95
8080A/8085 Assembly Language Programming	£6.95
Some Common BASIC Programs	£5.95
BUSINESS PROGRAMS IN BASIC	
Payroll With Cost Accounting	£9.95
Accounts Payable & Accounts Receivable (Available from March 78)	£9.95
General Ledger (Available Mar 78)	£9.95
From Scelbi Computer Consulting Inc.	
6800 Software Gourmet Guide & Cookbook	£7.95
8080 Software Gourmet Guide & Cookbook	£7.95
8080 Programmers Pocket Guide	£2.25
8080 Hex Code Card	£2.25
8080 Octal Code Card	£2.25
8080 Guide and One 8080 Code Card	£4.20
8080 Guide and Both Code Cards	£6.00
SCELBAL High Level Language for '8008/8080' Systems	£39.25
SCELBAL String Handling Supplement	£8.00
SCELBAL Extended Maths Supplement	£4.00
Understanding Microcomputers & Small Computer Systems	£7.95
SCELBI 'BYTE' Primer	£9.95
8080 Standard Assembler (In Book Format)	£15.95
From Peoples Computer Company	
Reference Book of Personal & Home Computing	£4.95
What to Do After You Hit Return	£7.00
Dr. Dobbs Journal Volume 1	£10.00
*** From Kilobaud/73 Magazine Inc.**	
Hobby Computers Are Here	£3.95
New Hobby Computers	£3.95
From Dymax Inc.	
Instant BASIC by Jerald R. Brown	£4.95
Your Home Computer by James White	£4.95
My Computer Likes Me . . . When I Speak BASIC By Bob Albrecht	£1.65
Games With A Pocket Calculator by Thiagarajan & Stilovitch	£1.75
Games, Tricks and Puzzles For a Hand Calculator by W Judd	£2.49
*** From BYTE Publications Inc.**	
Paperbytes:	
Tiny Assembler for 6800 Systems	£5.75
Bar Code Loader for 6800, 8080, Z80 & 6502 Micros	£1.75
Best of BYTE Volume 1	£8.95

		Price UK	Price Overseas If Different
*** From Creative Computing Press**			
Best of Creative Computing Volume 1		£ 6.95	
Best of Creative Computing Volume 2		£ 6.95	
101 BASIC Games (Revised & Reprinted Feb. 78)		£ 5.50	
The Colossal Computer Cartoon Book		£ 3.95	
Computer-Rage (A new Board Game)		£ 6.95	
Artist and Computer		£ 3.95	
Three Binary Dice		£ 1.00	
*** From Everyone Else**			
TV Typewriter Cookbook by Don Lancaster		£ 7.95	
Magazine storage boxes (Holds 12 minimum)		£ 1.75	
Sybex: Microprocessors		£ 7.95	
Sybex: Microprocessor Interfacing Techniques		£ 7.95	
Dilithium: Home Computers Volume 1: Hardware		£ 6.50	
Dilithium: Home Computers Volume 2: Software		£ 5.95	
Bugbooks: Volume 1 to 7		P.O.A.	
MAGAZINES: Back Issues			
Personal Computing		£ 1.75	
Interface Age		£ 2.00	
Dr. Dobbs Journal		£ 1.60	
Computer Music Journal		£ 2.50	
Peoples Computers		£ 1.50	
BYTE		£ 2.25	
Creative Computing		£ 1.75	
Calculators & Computers		£ 1.75	
ROM		£ 1.75	
Kilobaud		£ 2.25	
73		£ 2.00	
MAGAZINES: Subscriptions			
Personal Computing	(Twelve Issues Yearly)	£16.00	£17.00
Interface Age	(Twelve Issues Yearly)	£20.00	£20.50
Dr. Dobbs Journal	(Ten Issues Yearly)	£13.00	£13.50
Computer Music Journal	(Four Issues Yearly)	£ 8.50	£ 9.00
Peoples Computers	(Six Issues Yearly)	£ 8.00	£ 8.50
Kilobaud	(Twelve Issues Yearly)	£20.00	£21.00
BYTE	(Twelve Issues Yearly)	£15.00	£15.00
Creative Computing	(Six Issues Yearly)	£ 8.50	£ 9.00
Calculators & Computers	(Seven Issues Yearly)	£10.00	£10.50
ROM	(Twelve Issues Yearly)	£16.00	£17.00
73	(Twelve Issues Yearly)	£20.00	£21.00

THIS LIST CANCELS ALL PREVIOUS PRICE LISTS: EFFECTIVE FEBRUARY 1978 LPE/278/VC

HOW TO ORDER

Please note our prices include postage and packing. but not insurance, if wanted add 12p for every £10 of books ordered. Make cheques, PO's etc payable to:-

L. P. Enterprises

CREDIT CARDS accepted: BARCLAYCARD/VISA/-DINERS CLUB

Send to address above All Orders must be Prepaid

Indicate Payment Method:

.... My cheque, P.O., I.M.O. is enclosed in Sterling on U.K. Bank

.... Charge to Barclaycard/Visa/Diners

Number .... Expiry Date ....

Name ....

Address ....

.... POSTCODE ....

Signature ....

All publications are published in U.S.A. and shipped air-freight by L.P. Enterprises. In unusual cases, processing may exceed 30 days.

CORRECTIONS TO THE COMMODORE "PET COMPUTER USERS HANDBOOK"

Corrections should be made to the following pages of the above:-

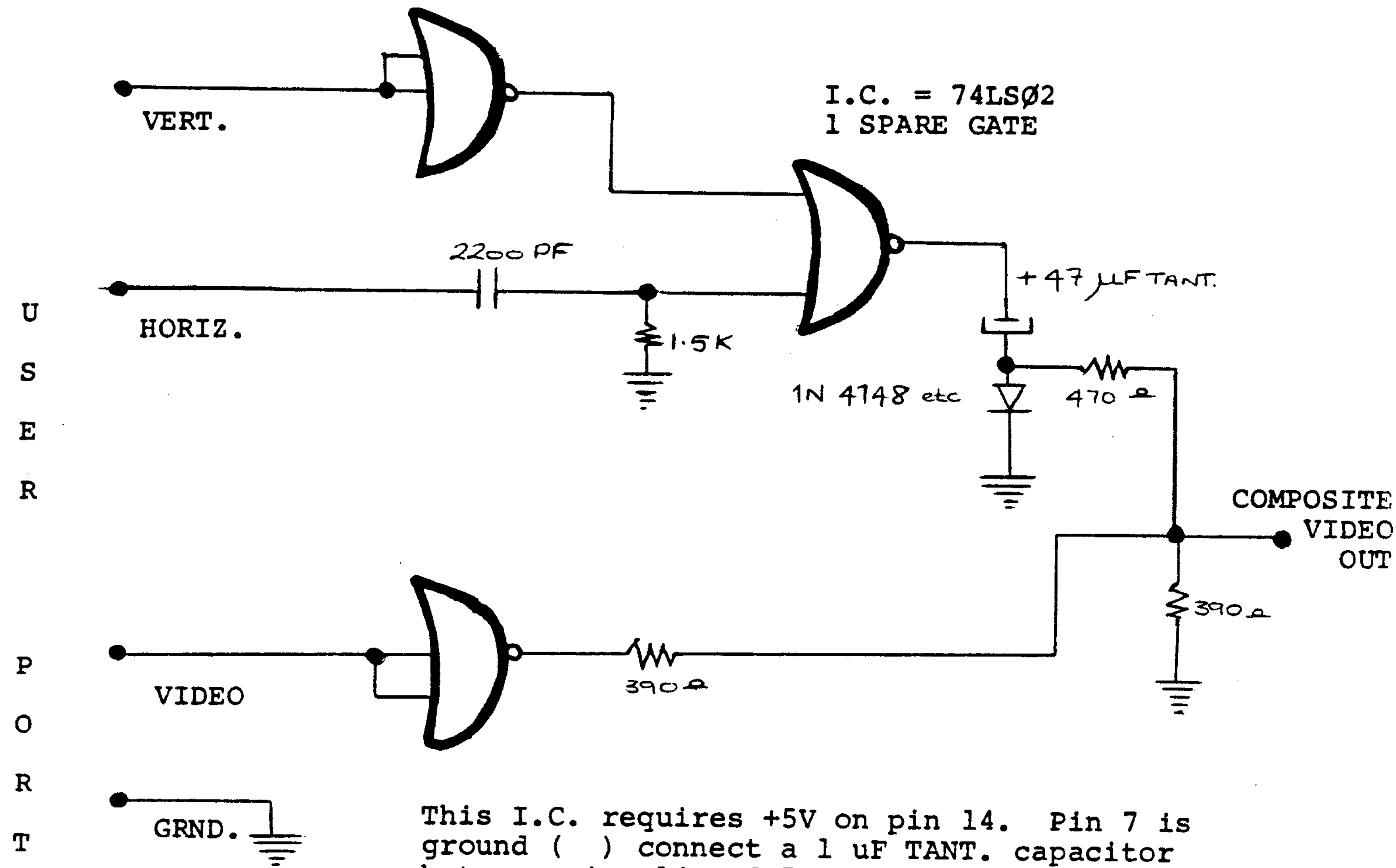
PAGE	LINE
16	3Ø NEXT IN - change to - 3Ø NEXT N
24	11Ø FOR I = 1 to 15 : READ A $ (1) : NEXT 1: REM READ STRINGS A $ (I) - change to - NEXT I:
24	13Ø IF A $ (1) = A$ (I+1) THEN 18Ø - change to - IF A $ (I) etc.
52	Delete lines 25Ø and 26Ø 4ØØ Place the ST outside the quotes.
69	1) Some BASICS use "["and"]" to denote matrix subscripts. PET BASIC uses "("and")".
92-100	omit
121	The address of R. Bailey Associates who supply memory and interfaces for the PET should be 31 Bassett Road, London W1Ø and not NW1Ø.

IEEE INFORMATION

We have had many enquiries concerning the exact operation of the IEEE bus particularly concerning timing. Commodore is now making available a document generated by Hewlett Packard which describes the IEEE-488 interface bus. This is a comprehensive and readable description of the BUS. The price of this document is £2.5Ø including postage and packing. (Cash with order please)

It is important to remember that Commodore has not implemented all of Hewlett Packard's suggestions. Please refer to the IEEE section of the Owners Handbook.

ATTACHING A VIDEO MONITOR TO PET



This I.C. requires +5V on pin 14. Pin 7 is ground ( ) connect a 1 uF TANT. capacitor between pins 14 and 7. observing polarity

Above is a simple circuit which takes the horizontal drive, vertical drive and video waveforms from the PET User Port and converts them to composite video suitable for driving an RF modulator or a straightforward monitor. The circuit requires a 5 volt power supply and this may be obtained from a 2nd cassette socket which has a few milliamps available at 5 volts. There are no particular points to watch out for when constructing this circuit. Lay-out is not critical. In the unlikely event of the horizontal hold of your display device misbehaving, adjust the value of the 1.5K resistor. This will alter the horizontal sync. pulse width.

DELAYS

Quite a few people have asked how to put delays into programs. Here are two common methods:

1Ø FOR A = 1 to 1ØØØ : NEXT this will cause a delay of approximately 1 second

1Ø FOR A = 1 to 2ØØØ : NEXT this will cause a delay of approximately 2 seconds etc.

```
1Ø T=TI
2Ø IF TI - T < 6Ø THEN 2Ø
```

Lines 1Ø and 2Ø cause a delay of approximately one second and work as follows:

Line 1Ø sets the variable T equal to the real time jiffy clock TI (a jiffy is 1/6Ø of a second)

Line 2Ø tests to see whether 6Ø/6Ø of a second have elapsed, if not the program returns to the beginning of line 2Ø and checks again.

Here is a small program you might like to try which uses delays involving the real time clock in an interesting manner.

```
READY.

 5 PRINT"KEY IN A NUMBER>";
 1Ø T=Ø:A$=""
 2Ø GETK$:IFK$=""THEN2Ø
 3Ø T=TI:GOTO6Ø
 4Ø GETK$
 5Ø IFTI-T>6ØTHEN7Ø
 6Ø IFK$<>""THENPRINTK$;:A$=A$+K$:T=TI:GOTO4Ø
 65 GOTO4Ø
 7Ø IFA=ØTHENPRINT"+";:A=VAL(A$):GOTO1Ø
 8Ø PRINT"="A+VAL(A$)
READY.
```

PLOTTING

It is possible, with very little effort, to address locations on the screen using simple XY co-ordinates. Below we have a program that uses a simple formula that enables one to do this.

```
READY.

 5 DATA12,15,22,5,12,25,33
 10 PRINT""
 20 PI=3.14159265
 30 FORA=0TO4*PI STEP(4*PI)/39
 40 Y=INT(SIN(A)*12+12):X=X+1
 50 GOSUB80
 60 NEXT
 70 FORA=33568TO33574:READZ:POKEA,Z:NEXT
 75 GOTO75
 80 POKE((24-Y)*40+32768)+X,46:RETURN
READY.
```

The line that does the actual XY co-ordinate conversion is line 8Ø. For the sake of clarity line 8Ø has been made a sub-routine but the formula is so compact that in some cases, including this one, it is not necessary. Line 5 and 7Ø should be included when you test this program out but may be omitted subsequently. X has a range of Ø-39 and Y has a range of Ø-24.

INPUTTING

It is worth pointing out that commas and colons act as delimiters in input strings, eg.

1Ø INPUT A $: ? A $ If this program is run and you type HOWEVER, I THINK the machine will accept HOWEVER and print the error message EXTRA IGNORED. The same will happen if you use the : in similar circumstances. If you wish to include either of these characters in an input statement enclose your typed INPUT in quotes. Many people must have been annoyed by the way BASIC will abort if the return key is pressed when the machine is waiting on an INPUT statement and no data has been typed in.

It is possible to arrange an input statement so that it will never do this. The method is as follows:

(note; → means CURSOR RIGHT and ← means CURSOR LEFT)

```
1Ø INPUT "→ → * ←← ←  ";A
```

When this input statement is encountered the user must type a number in reply, anything other than a number, including no entry at all, will cause the machine to return to the input statement with the appropriate message. Symbols other than * can be used where required.

DATA FILE ERRORS

There is a bug in the file handling routine which causes data to be written on the tape prematurely, not allowing for cassette motor start up time.

This is temporarily curable by keeping the motor running whilst the tape buffer is being filled, or by starting the motor when the buffer is almost filled.

The method of turning on the motor is to change a bit in the appropriate PIA register. The location of the PIA register is 59411 and the correct byte to place in that register is 53. Therefore the syntax for turning on the cassette motor is POKE 59411,53. This should be done either every time PRINT# is used or just before the buffer is full. Using the latter method involves PEEKING location 625 which is the buffer pointer. When this pointer approaches 191 which is the size of the buffer, turn on the motor. The relevant locations of bytes for the second cassette port are 59456 and 223 for STOP and 2Ø7 for START.

DATA FILE ERRORS (cont.)

A problem with opening files to write on either built-in cassette #1, or external cassette #2, has been discovered. When a file is opened, garbage will be written out instead of a proper data tape file header. Without this header, it is impossible to open the tape file for reading.

You may not have encountered this problem previously, because it is disguised by having loaded a program on the cassette prior to writing a data file. In this mode, the start address of the buffer with the header information is initialized properly but cassette data file operation still could be random.

Fortunately, there is a software patch you can implement in your BASIC program to force the open for write on tape to work every time.

Before opening to write on #1 cassette:

```
POKE 243,122
POKE 244,2
```

and on #2 cassette:

```
POKE 243,58
POKE 244,3
```

Locations 243 and 244 contain the lo and hi order bytes respectively of the address of the currently active cassette buffer. The start address of buffer #2 is $33A which is 3,58 ($3=3,$3A=58) in double byte decimal. Similarly cassette #1 is $27A ($2=2,$7A=122).

TAPE HEAD CARE

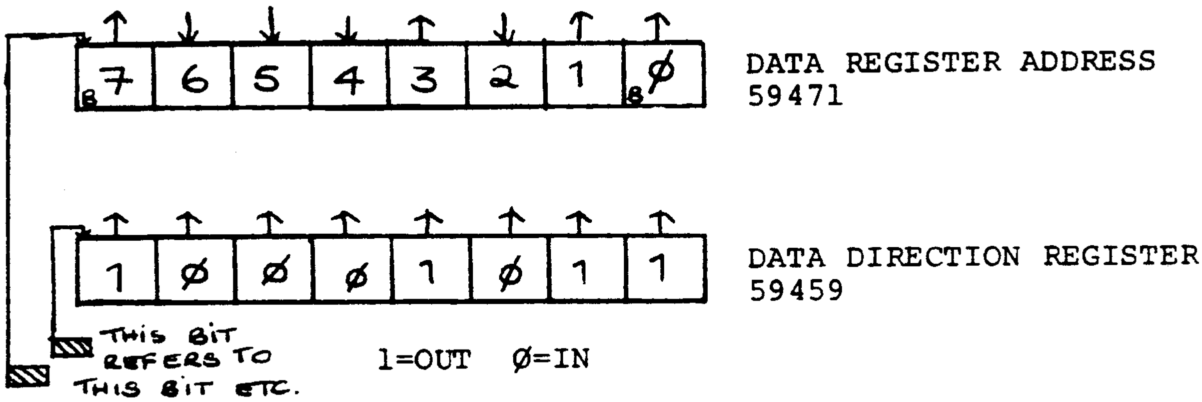
It has been noted that the READ/WRITE head in the PET cassette deck has the annoying habit of magnetising itself after a remarkably short period of operation. It is in fact possible to partially erase your tapes by up to 15% after only 15 or 2Ø passes over the head. The most convenient way to deal with this problem is to demagnestise the tape head very frequently, ie every couple of days with a demagnetising cassette. AMPEX market quite a good one for about £3.-

USEFUL ADDRESSES

On the following page you will find an extensive map of the PET memory. This list is "home" generated and not from CBM U.S. so may contain slight inaccuracies, but all the major buffers and ram areas are correct. Also here are some common PIA addresses and how to use them.

User Port - data register 59424

User Port Data Direction 59426



The major portion of the user port consists of 8 connections at the rear of the PET. Whether these connections are used for INPUT or OUTPUT is up to the programmer. These 8 wires may be used as either input or output. Before using this 8 bit port you must first configure these wires as inputs or outputs. This is done by writing a byte to the data direction register at address 59459. In the example above bits Ø, 1, 3 and 7 are configured as outputs. Bits 2,4,5 and 6 are configured as inputs. The bit that you see in the data direction register is generated by poke 59459, 139. In order to test a particular bit being used as an input in the data register

USEFUL ADDRESSES (CONT.)

(59471) one must peek 59471 and apply a "mask" in order to mask out unwanted bits. For instance to examine bit 2 we would use the expression PRINT PEEK (59471) AND 4. If the result of this expression is Ø then bit 2 of the data register (59471) has been held at Ø volts by the outside world.

PET MEMORY MAP

```
ØØØØ-ØØØ2   JUMP,USER ADDRESS
ØØØ5        CURSOR COLUMN
ØØØA-ØØ5A   BASIC INPUT BUFFER
ØØ5C        BASIC INPUT BUFFER POINTER
ØØ5E        CURRENT RESULT TYPE (FF) STRING (ØØ) NUMERIC
ØØ5F          "      "     "   (80) INTEGER(ØØ)FLOATING POINT
ØØ7A-ØØ7B   START OF BASIC STATEMENTS
ØØ7C-OO7D   START OF VARIABLE TABLE
ØØ7E-ØØ7F   END OF VARIABLE TABLE
ØØ8Ø-ØØ81   START OF AVAILABLE SPACE
ØØ82-ØØ83   BOTTOM OF STRINGS (MOVING DOWN)
ØØ84-ØØ85   TOP OF STRINGS (MOVING DOWN)
ØØ86-ØØ87   TOP OF MEMORY ALLOCATED FOR BASIC WORKING AREA
ØØ88-ØØ89   CURRENT PROGRAM LINE NUMBER
ØØ8A-ØØ88     "      "      "     "    SAVED BY END
ØØ8C-ØØ8D     "      "      POINTER SAVED BY END
ØØ92-ØØ93   DATA STATEMENT POINTER
ØØ94-ØØ95   CURRENT VARIABLE SYMBOLS
ØØ96-ØØ97   CURRENT VARIABLE STARTING POINT
ØØAE-ØØAF   POINTER ASSOCIATED WITH BASIC BUFF TRANSFER
ØØ8Ø        EXPONENT + S8Ø    )
ØØ81        MANTISSA      MSB )
ØØ82          "               )--(FLOATING POINT ACCUMULATOR)
ØØ83          "               )
ØØ84                      LSB )
ØØ85        SIGN OF MANTISSA (Ø IF ZERO)(+ IF POS.)(- IF NEG.)
ØØ88-ØØCØ   DYADIC HOLDING AREA
ØØØ2-       START OF ROUTINE FOR FETCHING NEXT BASIC CHARACTER
ØØC9-ØØCA   PROGRAM POINTER
    -ØØD9   END OF CHARACTER FETCH
ØØEØ-       SCREEN POSITION ON LINE
ØØE1-ØØE2   POSITION OF LINE START
ØØE3-ØØE4   CURRENT TAPE BUFFER POINTER
ØØE5-ØØE6   END OF CURRENT PROGRAM
ØØEA        QUOTE MODE (ØØ IF NOT IN QUOTE)
ØØEE        NUMBER OF CHARACTERS IN FILE NAME
```

ØØEF	GPIB FILE#
ØØFØ	GPIB COMMAND
ØØF1	GPIB DEVICE#
ØØF3-ØØF4	START OF TAPE BUFFER
ØØF5	CURRENT SCREEN LINE#
ØØF6	RUNNING CHECKSUM OF BUFFER
ØØF7-ØØF8	POINTER TO PROGRAM DURING VERIFY,LOAD
ØØF9-ØØFA	FILENAME STARTING POINTER
ØØFC	SERIAL WORD
ØØFD	NUMBER OF BLOCKS REMIATNING TO WRITE
ØØFE	SERIAL WORD BUFFER
ØØFF-1FF	BASIC STACK ETC.
Ø2ØØ-Ø2Ø2	CLOCK H.M.S.
Ø2Ø3	MATRIX COORDINATE OF LAST KEY DOWN(255 IF NONE)
Ø2Ø4	SHIFT KEY STATUS (1 IF DOWN)
Ø2Ø5-Ø2Ø6	JIFFY CLOCK
Ø2Ø7	CASSETTE 1 ON SWITCH
Ø2Ø8	CASSETTE 2 ON SWITCH
Ø2Ø9	KEYSWITCH PIA
Ø2ØB	LOAD O, VERIFY 1
Ø2ØC	STATUS
Ø2ØD	NUMBER OF CHR IN KBD BUFFER
Ø2ØE-Ø216	KYBD INPUT BUFFER
Ø219-Ø21A	HARDWARE INTERRUPT VECTOR
Ø21B-Ø21C	BREAD INTERRUPT VECTOR
Ø223	KEY IMAGE
Ø225	CURSOR TIMING
Ø228	TAPE WRITE
Ø242-Ø24B	LOGICAL NUMBERS OF OPEN FILES
Ø24C-Ø255	DEVICE NUMBERS OF OPEN FILES
Ø256-Ø25F	R/W MODES OF OPEN FILES (COMMAND TABLE)
Ø262	GPIB TABLE LENGTH
Ø265	PARITY
Ø268	POINTER IN FILENAME TRANSFER
Ø26C	SERIAL BIT COUNT
Ø27Ø	TAPE WRITE COUNTDOWN
Ø273	LEADER COUNTER

Ø275	Ø IF FIRST HALF BYTE MARKER NOT WRITTEN
Ø276	Ø IF SECOND " " " " "
Ø279	CHECKSUM WORKING WORD
Ø27A-Ø339	BUFFER FOR CASSETTE #1
Ø33A-Ø3F9	" " " #2
Ø4ØØ	START OF BASIC STATEMENTS
-1FFF	END OF AVAILABLE RAM (8K VERSION)
-7FFF	END OF AVAILABLE RAM EXPANSION
8ØØØ-8FFF	VIDEO RAM
9ØØØ-BFFF	AVAILABLE ROM EXPANSION AREA
CØØØ-EØBØ	MICROSOFT "8K" BASIC
EØ85-E27D	SYSTEM SET UP
E294-E66A	VIDEO DRIVER
E66B-E684	INTERRUPT HANDLER
E685-E75B	CLOCK UPDATE,KYBD SCAN(60HZ INT.)
E75C-E7D4	KYBD ENCODING TABLE
E8ØØ-EFFF	PIA'S
FØ86-F226	GPIB HANDLER
F346-F82C	FILE CONTROL
F82D-Fd15	TAPE CONTROL
FD38-FFB2	DIAGNOSTICS
FFCØ-FFED	JUMP VECTORS
FFFA-FFFF	65Ø2 INTERRUPT VECTORS (NMI NOT USED IN ORIG. VERSIONS)

MACHINE CODE ENVIRONMENT

If you wish to write machine code programs in your PET and do not wish to have BASIC trampling all over them here is a suggestion:

When the PET is first powered up a test pattern is written into and read back from the RAM in ascending address order. When this routine discovers a location which does not read back properly it presumes that it has run out of RAM and displays XXXX bytes free. At this point it makes a note of where it thinks the 'top of memory' is.. A quick glance at the memory map will show that BASIC program text is stored from location 1Ø25 upwards and strings are stored from the top of the memory downwards which means that in any normal circumstances there is nowhere in the PET main memory where you can hide your machine code routines.
If however, the first thing you do after powering up the PET is to alter the top of memory pointer to say 6ØØØ everything from 6ØØ1 upwards, as far as PET is concerned, does not exist. e.g. strings will be stored from 6ØØØ downwards etc. and machine code programs can be safely put in location 6ØØ1 upwards. This pointer is held in locations 134 and 135 constituting a 16 bit pointer with 134 being its lower 8 bits. This is a binary pointer which means that we must convert your 6ØØØ or whatever to binary before POKING locations 134 and 135 with the information. In the standard 8K PET 134 will be Ø and 135

MACHINE CODE ENVIRONMENT (cont.)

will be 32 (32 x 256 = 8192) Remember that 1Ø25 bytes are used for house keeping by the PET (8192 - 1Ø25 = 7167) However to give the PET a ceiling of 6ØØØ we convert 6ØØØ into binary which gives us POKE 134, 112 and POKE 135, 23.

LIFE FOR YOUR PET

Here is a good example of what can be done in machine code in the PET. It is the game of "LIFE" by John H. Conway of Cambridge. If one attempts to write a Commodore PET screen size (1ØØØ cell) version of LIFE in BASIC it can take up to two or three minutes per generation. This program performs two generations per second. In order to use it type in a listing in the form of data statements and load in the machine code with a small BASIC routine being careful to fill in the gaps between 1928 (HEX) and 193Ø and also 1954 and 197Ø with no-ops. Below is a listing of the documentation provided by the author.

LIFE FOR YOUR PET

Since this is the first time I have attempted to set down a machine language program for the public eye, I will attempt to be as complete as practical without overdoing it.

The programs I will document here are concerned with the game of "LIFE", and are written in 6502 machine language specifically for the PET 2001 (8K version). The principles apply to any 6502 system with graphic display capability, and can be debugged (as I did) on non-graphic systems such as the KIM-1.

The first I heard of LIFE was in Martin Gardner's "Recreational Mathematics" section in Scientific American, Oct-Nov 1970; Feb. 1971. As I understand it, the game was invented by John H. Conway, an English mathematician. In brief, LIFE is a "cellular automation" scheme, where the arena is a rectangular grid (ideally of infinite size). Each square in the grid is either occupied or unoccupied with "seeds", the fate of which are governed by relatively simple rules, i.e. the "facts of LIFE". The rules are: 1. A seed survives to the next generation if and only if it has two or three neighbors (right, left, up, down, and the four diagonally adjacent cells) otherwise it dies of loneliness or overcrowding, as the case may be. 2. A seed is born in a vacant cell on the next generation if it has exactly 3 neighbors.

With these simple rules, a surprisingly rich game results. The original Scientific American article, and several subsequent articles reveal many curious and surprising initial patterns and results. I understand that there even has been formed a LIFE group, complete with newsletter, although I have not personally seen it.

The game can of course be played manually on a piece of graph paper, but it is slow and prone to mistakes, which have usually disasterous effects on the final results. It would seem to be the ideal thing to put to a microprocessor with bare-bones graphics, since the rules are so simple and there are essentially no arithmetic operations involved, except for keeping track of addresses and locating neighbors.

As you know, the PET-2001 has an excellent BASIC interpreter, but as yet very little documentation on machine language operation. My first stab was to write a BASIC program, using the entire PET display as the arena (more about boundaries later), and the filled circle graphic display character as the seed. This worked just fine, except for one thing - it took about 2-1/2 minutes for the interpreter to go through one generation! I suppose I shouldn't have been surprised since the program has to check eight neighboring cells to determine the fate of a particular cell, and do this 1000 times to complete the entire generation (40x25 characters for the PET display).

The program following is a 6502 version of LIFE written for the PET. It needs to be POKE'd into the PET memory, since I have yet to see or discover a machine language monitor for the PET. I did it with a simple BASIC program and many DATA statements (taking up much more of the program memory space than the actual machine language program!). A routine for assembling, and saving on tape machine language programs on the PET is sorely needed.

The program is accessed by the SYS command, and takes advantage of the display monitor (cursor control) for inserting seeds, and clearing the arena. Without a serious attempt at maximizing for speed, the program takes about 1/2 second to go through an entire generation, about 300 times faster than the BASIC equivalent! Enough said about the efficiency of machine language programming versus BASIC interpreters?

BASIC is great for number crunching, where you can quickly compose your program and have plenty of time to await the results.

The program may be broken down into manageable chunks by subroutining. There follows a brief description of the salient features of each section:

MAIN (hex 1900)

In a fit of overcaution (since this was the first time I attempted to write a PET machine language program) you will notice the series of pushes at the beginning and pulls at the end. I decided to save all the internal registers on the stack in page 1, and also included the CLD (clear decimal mode) just in case. Then follows a series of subroutine calls to do the LIFE generation and display transfers. The zero page location, TIMES, is a counter to permit several loops through LIFE before returning. As set up, TIMES is initialized to zero (hex location 1953) so that it will loop 256 times before jumping back. This of course can be changed either initially or while in BASIC via the POKE command. The return via the JMP BASIC (4C 8B C3) may not be strictly orthodox, but it seems to work all right.

INIT (hex 1930) and DATA (hex 193B)

This shorty reads in the constants needed, and stores them in page zero. SCR refers to the PET screen, TEMP is a temporary working area to hold the new generation as it is evolved, and RCS is essentially a copy of the PET screen data, which I found to be necessary to avoid "snow" on the screen during read/write operations directly on the screen locations. Up, down, etc. are the offsets to be added or subtracted from an address to get all the neighbor addresses. The observant reader will note the gap in the addresses between some of the routines.

TMPSCR (hex 1970)

This subroutine quickly transfers the contents of Temp and dumps it to the screen, using a dot (81 dec) symbol for a live cell (a 1 in TEMP) and a space (32 dec) for the absence of a live cell (a 0 in TEMP).

SCRTMP (hex 198A)

This is the inverse of TMPSCR, quickly transferring (and encoding) data from the screen into TEMP.

RSTORE (hex 19A6)

This subroutine fetches the initial addresses (high and low) for the SCR, TEMP, and RCS memory spaces.

NXTADR (hex 19BD)

Since we are dealing with 1000 bytes of data, we need a routine to increment to the next location, check for page crossing (adding 1 to the high address when it occurs), and checking for the end. The end is signaled by returning a 01 in the accumulator, otherwise a 00 is returned via the accumulator.

TMPRCS (hex 19E6)

The RCS address space is a copy of the screen, used as mentioned before to avoid constant "snow" on the screen if the screen were being continually accessed. This subroutine dumps data from TEMP, where the new generation has been computed, to RCS.

GENER (hex 1A00)

We finally arrive at a subroutine where LIFE is actually generated. After finding out the number of neighbors of the current RCS data byte from NBRS, GENER checks for births (CMPIM $03 at hex addr. 1A0E) if the cell was previously unoccupied. If a birth does not occur, there is an immediate branch to GENADR (the data byte remains 00). If the cell was occupied (CMPIM 81 dec at hex 1A08), OCC checks for survival (CMPIM $03 at hex 1A1A and CMPIM $02 at hex 1A1E), branching to GENADR when these two conditions are met, otherwise the cell dies (LDAIM $00 at hex 1A22). The results are stored in TEMP for the 1000 cells.

NBRS (hex 1A2F)

NBRS is the subroutine that really does most of the work and where most of the speed could be gained by more efficient programming. Its job, to find the total number of occupied neighbors of a given RCS data location, is complicated by page crossing and edge boundaries. In the present version, page crossing is taken care of, but edge boundaries (left, right, top, and bottom of the screen) are somewhat "strange". Above the top line and below the bottom line are considered as sort of forbidden regions where there should practically always be no "life" (data in those regions are not defined by the program, but I have found that there has never been a case where 81's have been present (all other data is considered as "unoccupied" characters). The right and left edges are different, however,

and lead to a special type of "geometry". A cell at either edge is not considered as special by NBRS, and so to the right of a right-edge location is the next sequential address. On the screen this is really the left edge location, and one line lower. The inverse is true, of course for left addresses of left-edge locations. Topologically, this is equivalent to a "helix". No special effects of this are seen during a simple LIFE evolution since it just gives the impression of disappearing off one edge while appearing on the other edge. For an object like the "spaceship" (see Scientific American articles), then, the path eventually would cover the whole LIFE arena. The fun comes in when a configuration spreads out so much that it spills over both edges, and interacts with itself. This, of course cannot happen in an infinite universe, so that some of the more complex patterns will not have the same fate in the present version of LIFE. Most of the "blinkers", including the "glider gun" come out OK.

This 40x25 version of LIFE can undoubtedly be made more efficient, and other edge algorithms could be found, but I chose to leave it in its original form as a benchmark for my first successfully executed program in writing machine language on the PET. One confession, however - I used the KIM-1 to debug most of the subroutines. Almost all of them did not run on the first shot! Without a good understanding of PET memory allocation particularly in page zero, I was bound to crash many times over, with no recovery other than pulling the plug. The actual BASIC program consisted of a POKING loop with many DATA statements (always save on tape before running!).

A Brief Introduction
to the Game of Life

One of the interesting properties of the game of LIFE is that such simple rules can lead to such complex activity. The simplicity comes from the fact that the rules apply to each individual cell. The complexity comes from the interactions between the individual cells. Each individual cell is affected by its eight adjacent neighbors, and nothing else.

The rules are:

1. A cell survives if it has two or three neighbors.

2. A cell dies from overcrowding if it has four or more neighbors. It dies from isolation if it has one or zero neighbors.

3. A cell is born when an empty space has exactly three neighbors.

With these few rules, many different types of activity can occur. Some patterns are STABLE, that is they do not change at all. Some are REPEATERS, patterns which undergo one or more changes and return to the original pattern. A REPEATER may repeat as fast as every other generation, or may have a longer period. A GLIDER is a pattern which moves as it repeats.

REPEATERS

```
    * *         *
    * *           * *       * * *
* *           * *         * * *
* *             *
```

STABLE

```
          *
* *     *   *     *   * *
* *     *   *     * *   *
          *
```

GLIDERS

```
                  *
  *                 *
    *       *       *
* * *         * * * * * *
```

```
1900            LIFE   ORG    $1900

1900            BASIC  *      $C38B  RETURN TO BASIC ADDRESS
1900            OFFSET *      $002A  PAGE ZERO DATA AREA POINTER
1900            DOT    *      $0051  DOT SYMBOL = 81 DECIMAL
1900            BLANK  *      $0020  BLANK SYMBOL = 32 DECIMAL

1900            SCRL   *      $0020  PAGE ZERO LOCATIONS
1900            SCRH   *      $0021
1900            CHL    *      $0022
1900            CHH    *      $0023
1900            SCRLO  *      $0024
1900            SCRHO  *      $0025
1900            TEMPL  *      $0026
1900            TEMPH  *      $0027
1900            TEMPLO *      $0028
1900            TEMPHO *      $0029
1900            UP     *      $002A
1900            DOWN   *      $002B
1900            RIGHT  *      $002C
1900            LEFT   *      $002D
1900            UR     *      $002E
1900            UL     *      $002F
1900            LR     *      $0030
1900            LL     *      $0031
1900            N      *      $0032
1900            SCRLL  *      $0033
1900            SCRLH  *      $0034
1900            RCSLO  *      $0035
1900            RCSHO  *      $0036
1900            TMP    *      $0037
1900            TIMES  *      $0038
1900            RCSL   *      $0039
1900            RCSH   *      $003A

1900 08         MAIN   PHP           SAVE EVERYTHING
1901 48                PHA           ON STACK
1902 8A                TXA
1903 48                PHA
1904 98                TYA
1905 48                PHA
1906 BA                TSX
1907 8A                TXA
1908 48                PHA
1909 D8                CLD           CLEAR DECIMAL MODE
190A 20 30 19          JSR    INIT
190D 20 8A 19          JSR    SCRTMP
1910 20 E6 19   GEN    JSR    TMPRCS
1913 20 00 1A          JSR    GENER
1916 20 70 19          JSR    TMPSCR
1919 E6 38             INCZ   TIMES  REPEAT 255 TIMES
191B D0 F3             BNE    GEN    BEFORE QUITTING
191D 68                PLA           RESTORE EVERYTHING
191E AA                TAX
191F 9A                TXS
1920 68                PLA
```

```
1921 A8                 TAY
1922 68                 PLA
1923 AA                 TAX
1924 68                 PLA
1925 28                 PLP
1926 4C 8B C3           JMP   BASIC   RETURN TO BASIC

1930                    ORG   $1930

              MOVE VALUES INTO PAGE ZERO

1930 A2 19     INIT     LDXIM $19     MOVE 25. VALUES
1932 BD 3A 19  LOAD     LDAX  DATA    -01
1935 95 1F              STAZX $1F     STORE IN PAGE ZERO
1937 CA                 DEX
1938 D0 F8              BNE   LOAD
193A 60                 RTS

193B 00        DATA     =     $00     SCRL
193C 80                 =     $80     SCRH
193D 00                 =     $00     CHL
193E 15                 =     $15     CHH
193F 00                 =     $00     SCRLO
1940 80                 =     $80     SCRHO
1941 00                 =     $00     TEMPL
1942 1B                 =     $1B     TEMPH
1943 00                 =     $00     TEMPLO
1944 1B                 =     $1B     TEMPHO
1945 D7                 =     $D7     UP
1946 28                 =     $28     DOWN
1947 01                 =     $01     RIGHT
1948 FE                 =     $FE     LEFT
1949 D8                 =     $D8     UR
194A D6                 =     $D6     UL
194B 29                 =     $29     LR
194C 27                 =     $27     LL
194D 00                 =     $00     N
194E E8                 =     $E8     SCRLL
194F 83                 =     $83     SCRLH
1950 00                 =     $00     RCSLO
1951 15                 =     $15     RCSHO
1952 00                 =     $00     TMP
1953 00                 =     $00     TIMES

1970                    ORG   $1970

1970 20 A6 19  TMPSCR   JSR   RSTORE  GET INIT ADDRESSES
1973 B1 26     TSLOAD   LDAIY TEMPL   FETCH BYTE FROM TEMP
1975 D0 06              BNE   TSONE   BRANCH IF NOT ZERO
1977 A9 20              LDAIM BLANK   BLANK SYMBOL
1979 91 20              STAIY SCRL    DUMP IT TO SCREEN
197B D0 04              BNE   TSNEXT
197D A9 51     TSONE    LDAIM DOT     DOT SYMBOL
197F 91 20              STAIY SCRL    DUMP IT TO SCREEN
1981 20 BD 19  TSNEXT   JSR   NXTADR  FETCH NEXT ADDRESS
1984 F0 ED              BEQ   TSLOAD
```

```
1986 20 A6 19         JSR   RSTORE RESTORE INIT ADDRESSES
1989 60               RTS

198A 20 A6 19  SCRTMP JSR   RSTORE GET INIT ADDRESSES
198D B1 20     STLOAD LDAIY SCRL   READ DATA FROM SCREEN
198F C9 51            CMPIM DOT    TEST FOR DOT
1991 F0 06            BEQ   STONE  BRANCH IF DOT
1993 A9 00            LDAIM $00    OTHERWISE ITS A BLANK
1995 91 26            STAIY TEMPL  STORE IT
1997 F0 04            BEQ   STNEXT UNCOND. BRANCH
1999 A9 01     STONE  LDAIM $01    A DOT WAS FOUND
199B 91 26            STAIY TEMPL  STORE IT
199D 20 BD 19  STNEXT JSR   NXTADR FETCH NEXT ADDRESS
19A0 F0 EB            BEQ   STLOAD
19A2 20 A6 19         JSR   RSTORE RESTORE INIT ADDRESSES
19A5 60               RTS

19A6 A9 00     RSTORE LDAIM $00    ZERO A, X, Y
19A8 AA               TAX
19A9 A8               TAY
19AA 85 20            STAZ  SCRL   INIT VALUES
19AC 85 26            STAZ  TEMPL
19AE 85 39            STAZ  RCSL
19B0 A5 25            LDAZ  SCRHO
19B2 85 21            STAZ  SCRH
19B4 A5 29            LDAZ  TEMPHO
19B6 85 27            STAZ  TEMPH
19B8 A5 36            LDAZ  RCSHO
19BA 85 3A            STAZ  RCSH
19BC 60               RTS

19BD E6 26     NXTADR INCZ  TEMPL  GET NEXT LOW ORDER
19BF E6 20            INCZ  SCRL   BYTE ADDRESS
19C1 E6 39            INCZ  RCSL
19C3 E8               INX
19C4 E4 33            CPXZ  SCRLL  IS IT THE LAST?
19C6 F0 0C            BEQ   PAGECH IS IT THE LAST PAGE?
19C8 E0 00            CPXIM $00    IS IT A PAGE BOUNDARY?
19CA D0 0E            BNE   NALOAD IF NOT, THEN NOT DONE
19CC E6 27            INCZ  TEMPH  OTHERWISE ADVANCE TO
19CE E6 21            INCZ  SCRH   NEXT PAGE
19D0 E6 3A            INCZ  RCSH
19D2 D0 06            BNE   NALOAD UNCONDITIONAL BRANCH
19D4 A5 34     PAGECH LDAZ  SCRLH  CHECK FOR LAST PAGE
19D6 C5 21            CMPZ  SCRH
19D8 F0 03            BEQ   NADONE IF YES, THEN DONE
19DA A9 00     NALOAD LDAIM $00    RETURN WITH A=0
19DC 60               RTS
19DD A9 01     NADONE LDAIM $01    RETURN WITH A=1
19DF 60               RTS

19E6                  ORG   $19E6

19E6 20 A6 19  TMPRCS JSR   RSTORE INIT ADDRESSES
19E9 B1 26     TRLOAD LDAIY TEMPL  FETCH DATA FROM TEMP
19EB D0 06            BNE   TRONE  IF NOT ZERO THEN ITS ALIVE
```

```
19ED A9 20              LDAIM BLANK  BLANK SYMBOL
19EF 91 39              STAIY RCSL   STORE IT IN SCREEN COPY
19F1 D0 04              BNE   NEWADR THEN ON TO A NEW ADDRESS
19F3 A9 51       TRONE  LDAIM DOT    THE DOT SYMBOL
19F5 91 39              STAIY RCSL   STORE IT IN SCREEN COPY
19F7 20 BD 19    NEWADR JSR   NXTADR FETCH NEXT ADDRESS
19FA F0 ED              BEQ   TRLOAD IF A=0, THEN NOT DONE
19FC 20 A6 19           JSR   RSTORE ELSE DONE. RESTORE
19FF 60                 RTS

1A00 20 A6 19    GENER  JSR   RSTORE INIT ADDRESSES
1A03 20 2F 1A    AGAIN  JSR   NBRS   FETCH NUMBER OF NEIGHBORS
1A06 B1 39              LDAIY RCSL   FETCH CURRENT DATA
1A08 C9 51              CMPIM DOT    IS IT A DOT?
1A0A F0 0C              BEQ   OCC    IF YES, THEN BRANCH
1A0C A5 32              LDAZ  N      OTHERWISE ITS BLANK
1A0E C9 03              CMPIM $03    SO WE CHECK FOR
1A10 D0 14              BNE   GENADR A BIRTH
1A12 A9 01       BIRTH  LDAIM $01    IT GIVES BIRTH
1A14 91 26              STAIY TEMPL  STORE IT IN TEMP
1A16 D0 0E              BNE   GENADR INCONDITIONAL BRANCH
1A18 A5 32       OCC    LDAZ  N      FETCH NUMBER OF NEIGHBORS
1A1A C9 03              CMPIM $03    IF IT HAS 3 OR 2
1A1C F0 08              BEQ   GENADR NEIGHBORS IT SURVIVES
1A1E C9 02              CMPIM $02
1A20 F0 04              BEQ   GENADR
1A22 A9 00       DEATH  LDAIM $00    IT DIED!
1A24 91 26              STAIY TEMPL  STORE IT IN TEMP
1A26 20 BD 19    GENADR JSR   NXTADR FETCH NEXT ADDRESS
1A29 F0 D8              BEQ   AGAIN  IF 0, THEN NOT DONE
1A2B 20 A6 19           JSR   RSTORE RESTORE INIT ADDRESSES
1A2E 60                 RTS

1A2F 98          NBRS   TYA          SAVE Y AND X ON STACK
1A30 48                 PHA
1A31 8A                 TXA
1A32 48                 PHA
1A33 A0 00              LDYIM $00    SET Y AND N = 0
1A35 84 32              STYZ  N
1A37 A2 08              LDXIM $08    CHECK 8 NEIGHBORS
1A39 B5 29       OFFS   LDAZX OFFSET -01
1A3B 10 15              BPL   ADD    ADD IF OFFSET IS POSITIVE
1A3D 49 FF              EORIM $FF    OTHERWISE GET SET TO
1A3F 85 37              STAZ  TMP    SUBTRACT
1A41 38                 SEC          SET CARRY BIT FOR SUBTRACT
1A42 A5 39              LDAZ  RCSL
1A44 E5 37              SBCZ  TMP    SUBTRACT TO GET THE
1A46 85 22              STAZ  CHL    CORRECT NEIGHBOR ADDRESS
1A48 A5 3A              LDAZ  RCSH
1A4A 85 23              STAZ  CHH
1A4C B0 11              BCS   EXAM   OK, FIND OUT WHAT'S THERE
1A4E C6 23              DECZ  CHH    PAGE CROSS
1A50 D0 0D              BNE   EXAM   UNCOND. BRANCH
1A52 18          ADD    CLC          GET SET TO ADD
1A53 65 39              ADCZ  RCSL   ADD
1A55 85 22              STAZ  CHL    STORE THE LOW PART
```

```
1A57 A5 3A          LDAZ  RCSH   FETCH THE HIGH PART
1A59 85 23          STAZ  CHH
1A5B 90 02          BCC   EXAM   OK, WHAT'S THERE
1A5D E6 23          INCZ  CHH    PAGE CROSSING
1A5F B1 22   EXAM   LDAIY CHL    FETCH THE NEIGHBOR
1A61 C9 51          CMPIM DOT    DATA BYTE AND SEE IF ITS
1A63 D0 02          BNE   NEXT   OCCUPIED
1A65 E6 32          INCZ  N      ACCUMULATE NUMBER OF NEIGHBORS
1A67 CA      NEXT   DEX
1A68 D0 CF          BNE   OFFS   NOT DONE
1A6A 68             PLA          RESTORE X, Y FROM STACK
1A6B AA             TAX
1A6C 68             PLA
1A6D A8             TAY
1A6E 60             RTS
```

This program was prepared by:

Dr. F. H. Covitz,
Deer Hill Road,
Lebanon,
N.J. 08833,
USA.

LIFE FOR YOUR PET (cont.)

Below we have a way of actually getting our HEX OP-CODES into the PET. Lines 1ØØ-2ØØ read the data statements convert them to decimal and POKE them sequentially into the memory. The first data item is expected to be the starting point of the loading in decimal and the last data item is expected to be an asterix. The beauty of this method is that you can use the screen edit facility on the PET for inserting and deleting codes. When you have inserted your own data statements from line 3ØØ upwards, save the entire performance prior to running as machine language routines rarely work first time around and the PET is quite likely to hang up and need turning off and on. The data statements in the example are for the game of LIFE. In the original version listed on the previous pages, 256 generations must occur before the control returns to BASIC. I have modified the program slightly in the beginning in order to allow the stop button to halt the binary routine. If you think you have loaded the following program correctly type RUN and press RETURN. This loads the binary program. When the machine prints READY, clear the screen. Type say eight shifted Qs in a row in the middle of the screen followed by SYS (64ØØ) (which is 19ØØH in decimal) and press return. GOOD LUCK!

```
100 READL
110 READ A$:C=LEN(A$):IFA$="*"THENEND
120 IFC<1ORC>2THEN200
130 A=ASC(A$)-48:B=ASC(RIGHT$(A$,1))-48
140 N=B+7*(B>9)-(C=2)*(16*(A+7*(A>9)))
150 IFN<0ORN>255THEN200
160 POKEL,N:L=L+1:GOTO110
200 PRINT"BYTE"L"=["A$"] ???":END
300 DATA6400
310 DATA 08,48,8A,48,98,48,BA,8A,48,D8,20,30,19,20,8A,19,20,E6,19,20,00,1A
320 DATA20,70,19,A9,FF,CD,12,E8,F0,F0,4C,8B,C3,AA,68,28,4C,8B,C3
330 DATA EA,EA,EA,EA,EA,EA,EA,A2,19,BD,3A,19,95,1F,CA,D0,F8,60,00,80,00,15,00
340 DATA80,00,1B,00,1B,D7,28,01,FE,D8,D6,29,27,00,E8,83,00,15,00,00
350 DATAEA,EA,EA,EA,EA,EA,EA,EA,EA,EA,EA,EA,EA,EA,EA,EA,EA,EA,EA,EA,EA,EA,EA
360 DATAEA,EA,EA,EA,EA,20,A6,19,B1,26,D0,06,A9,20,91,20,D0,04,A9,51,91,20,20
370 DATA BD,19,F0,ED,20,A6,19,60,20,A6,19,B1,20,C9,51,F0,06,A9,00,91,26,F0
380 DATA04,A9,01,91,26,20,BD,19,F0,EB,20,A6,19,60,A9,00,AA,A8,85,20,85,26,85
390 DATA39,A5,25,85,21,A5,29,85,27,A5,36,85,3A,60,E6,26,E6,20,E6,39,E8,E4
400 DATA33,F0,0C,E0,00,D0,0E,E6,27,E6,21,E6,3A,D0,06,A5,34,C5,21,F0,03,A9,00
410 DATA 60,A9,01,60,EA,EA,EA,EA,EA,EA,20,A6,19,B1,26,D0,06,A9,20,91,39,D0
420 DATA04,A9,51,91,39,20,BD,19,F0,ED,20,A6,19,60,20,A6,19,20,2F,1A,B1,39,C9
430 DATA51,F0,0C,A5,32,C9,03,D0,14,A9,01,91,26,D0,0E,A5,32,C9,03,F0,08,C9,02
440 DATAF0,04,A9,00,91,26,20,BD,19,F0,D8,20,A6,19,60,98,48,8A,48,A0,00,84,32
450 DATAA2,08,B5,29,10,15,49,FF,85,37,38,A5,39,E5,37,85,22,A5,3A,85,23,B0,11
460 DATAC6,23,D0,0D,18,65,39,85,22,A5,3A,85,23,90,02,E6,23,B1,22,C9,51,D0,02
470 DATAE6,32,CA,D0,CF,68,AA,68,A8,60,*
READY.
```

YOUR LETTERS

M.P. Glynne (B.Sc) of Tarn Cottage, Tarn Lane, Shadwell, Leeds wrote in to say:

Othello Program for P.E.T. - there is a bug in the above which has the following effect; the computer forfeits moves when, in fact, it has legal moves available. This arises because statements 1215 and 1220 assign a negative weighted value of -1 to locations 2D, 2E, 4B, 4G, 5B, 5G, 7D, 7E and -2 to 2B, 2G, 7B, 7G. If Sub 2820 identifies only one piece to capture then at statement 1200 S1 = 1. After weighting S1 will be assigned 0 or -1. Taking the '0' case, then B1 will be assigned S1 = 0 (statement 1340) and then 1410 will not be true, and 1430 "I have to forfeit my move", will result.

To rectify this completely for both the -1 and the -2 weighted cases the following amendment will be necessary;-

Statement 1000 'B1 = -1' to be amended to 'B1 = -2:

Statement 1410 to be amended to 'If B1 = -1 then 1480

Mr. J. Smith of 38 Claremont Crescent, Croxley Green, Rickmansworth, Herts. WD3 3QR

wrote in: The error in the definition of arc cos X should, I feel, be corrected. A possible version is:- (*)

ACS X = ATN(SQR (1-X↑2)/X) + (1-SGN(X))*π/2

this correctly gives (<u>unless</u> <u>X=0</u>) arc cos (-0.5) as

Cont/...

YOUR LETTERS (cont.)

$2\pi/3$ ($12\emptyset^o$) ; your formula gives

arc cos ($-\emptyset.5$) as $-6\emptyset^o$ this would be incorrect in e.g. a "cosine rule" problem.

As you expect PET to be used in educational establishments for solving trig. problems, I think it important to put this right.

(*) Note that if X is <u>negative</u>

1-SGN(X) = 2

& if X is <u>positive</u>

1-SGN(X) = Ø

this ensures that a correct multiple of π is added to the arctangent. Also, would it not be better to suggest..

P = $18\emptyset/\pi$ (<u>before</u> FNS is used)

DEFFNS(V) = SIN (V/P) etc.

for the user defined functions?

HERE ARE SOME COMMENTS FROM MR. M.J. SMYTH who is ~~the~~ Senior Lecturer, Department of Astronomy, Royal Observatory, Edinburgh EH9 3HJ.

Using BASIC and the IEEE 488 bus, PET can input 4Ø numbers per second from a 3½ digit voltmeter (Hewlett Packard 3437A). Also using BASIC, the user port can generate an output trigger (e.g. to a measuring device)

YOUR LETTERS (cont.)

within about 1Ø ms of an input trigger. We have not yet tried using assembler. But the BASIC speeds make possible very interesting applications in equipment control and real-time data processing.

RETAIL PRICE LIST

PET MEMORY

READY TO PLUG IN AND GO. EACH MEMORY MODULE IS A BOXED UNIT COMPLETE WITH POWER SUPPLY AND APPROPRIATE CONNECTORS.AN ALTERNATIVE MODEL WHICH MOUNTS INSIDE THE PET AND REQUIRES NO EXTRA POWER SUPPLY IS AVAILABLE.

SIZE	RRP	5 UP LESS 12%
16 K	720	633.60
24 K	758	667.04

DELIVERY TO COMMENCE JULY

IEEE/RS232C SERIAL INTERFACE 'A'

THIS IS A UNIDIRECTIONAL INTERFACE SUITABLE FOR ANY PRINTER OR SERIAL DEVICE REQUIRING V24/RS232C OR CURRENT LOOP SIGNALS. THE INTERFACE MAY BE USED WITH UP TO 15 OTHER DEVICES SIMULTANEOUSLY CONNECTED TO THE IEEE BUS. THE UNIT CONTAINS SPECIAL FORMATTING CIRCUITRY FOR CONVERTING PET CODES TO ASCII. THE UNIT IS NORMALLY SUPPLIED WIRED FOR LISTEN ADDRESS 4.

THE INTERFACE IS SUPPLIED COMPLETE WITH POWER SUPPLY AND IS HOUSED IN A SMALL INSTRUMENT CASE. A CABLE AND EDGE CONNECTOR ARE PROVIDED FOR THE PET. CONNECTION TO THE SERIAL DEVICE IS VIA A STANDARD 25 WAY D-TYPE CONNECTOR.

BAUD RATES OF 110, 300, 600, 1200 ARE SELECTABLE BY MEANS OF A D.I.L. SWITCH. STOP BITS ARE ALSO SWITCH SELECTABLE. THE UNIT IS NORMALLY SET TO GIVE EVEN PARITY ALL BAUD RATE TIMING IS CRYSTAL CONTROLLED.

PRICE 106.00 DELIVERY 21/40 DAYS

SERIAL INTERFACE 'B'

THIS IS A SOPHISTICATED BIDIRECTIONAL INTERFACE WITH FULL LISTENER AND TALKER ADDRESS DECODING. THIS UNIT MAY BE USED WHERE REMOTE INPUT IS ALSO REQUIRED. OTHER SPECIFICATIONS AS FOR INTERFACE 'A'

PRICE 186.67 DELIVERY TO COMMENCE JULY

TERMS

ALL PRICES EX VAT AT 8%. ALL ORDERS MUST BE CWO
CHEQUES SHOULD BE MADE PAYABLE TO R. BAILEY ASSOCIATES.
ORDERS FOR INTERFACES SHOULD
INCLUDE 2.50 P&P PER UNIT.
ALL GOODS SUPPLIED UNDER 90 DAY WARRANTY.